PROJET AGORA

Utilisation du Framework Symfony

Mission 4

Utilisation du framework Symfony

Les besoins de la MJC AGORA s'accroissant, l'application AgoraBo sera progressivement complétée avec de nouvelles fonctionnalités. Afin d'en faciliter la maintenabilité et l'évolutivité, il a été décidé de refondre l'application pour utiliser le framework Symfony.

Présentation

Symfony est un ensemble de composants PHP, un framework d'application Web, une philosophie et une communauté - tous travaillant ensemble en harmonie.

Symfony est un des framework les plus prisés du web, qui, plus qu'un simple support de travail, est une technologie alliant rapidité de développement, sécurité, modularité et évolutivité du code. C'est un framework PHP open Source avec une architecture MVC (Model, View, Controller). Il permet aux développeurs de créer des applications web sans codage monotone et étendu. Grâce à ce framework, ils peuvent se servir d'outils pour réaliser un travail normé et organisé, poser les bases d'un projet web, suivre une architecture et une logique de travail strictes et rigoureuses. Cet applicatif est source de productivité pour un développeur car il dispose de "composants" qu'il est libre d'utiliser ou non dans son travail de développement web.

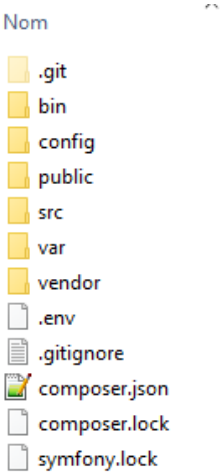
Principales étapes de la mise en œuvre de Symfony

Création du squelette de l'application Symfony

Dans une fenêtre de commande, positionné sur la racine web www, il faut saisir la commande : symfony new AgoraBoS

```
C:\wamp64\www>symfony new AgoraBoS

INFO A new Symfony CLI version is available (5.4.19, currently running 5.4.18).

     If you installed the Symfony CLI via a package manager, updates are going to be automatic.
     If not, upgrade by downloading the new version at https://github.com/symfony-cli/symfony-cli/releases
     And replace the current binary (symfony.exe) by the new one.

* Creating a new Symfony project with Composer
  (running C:\ProgramData\ComposerSetup\bin\composer.phar create-project symfony/skeleton C:\wamp64\www\AgoraBoS  --no-interaction)

* Setting up the project under Git version control
  (running git init C:\wamp64\www\AgoraBoS)

[OK] Your project is now ready in C:\wamp64\www\AgoraBoS


C:\wamp64\www>
```

Un dossier AgoraBoS a été ajouté avec le contenu ci-dessous.

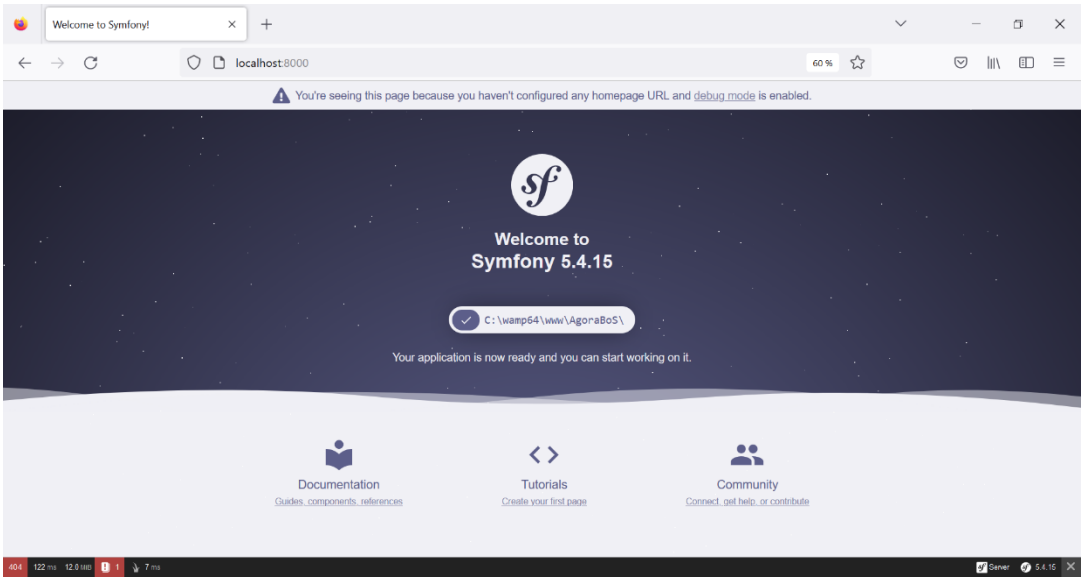


Démarrage du serveur et de l'application

Un serveur web est intégré à Symfony depuis la version 5.4 de PHP. Ainsi on est sûr que le serveur utilisera la même version de PHP que celle utilisée en ligne de commande. Pour le démarrer, il faut lancer la commande : symfony serve –no-tls

```
C:\wamp64\www\AgoraBoS>symfony serve --no-tls

INFO A new Symfony CLI version is available (5.4.19, currently running 5.4.18).

      If you installed the Symfony CLI via a package manager, updates are going to be automatic.
      If not, upgrade by downloading the new version at https://github.com/symfony-cli/symfony-cli/releases
      And replace the current binary (symfony.exe) by the new one.

Following Web Server log file (C:\Users\delio\.symfony5\log\e22671e67880eeecfbf1f81dec052965ca49763e.log)
Following PHP-CGI log file (C:\Users\delio\.symfony5\log\e22671e67880eeecfbf1f81dec052965ca49763e\79ca75f9e90b4126a5955a33ea6a41ec5e854698.log)

 [WARNING] The local web server is optimized for local development and MUST never be used in a production setup.

 [OK] Web server listening
      The Web server is using PHP CGI 7.4.9
      http://127.0.0.1:8000

[Web Server ] Nov 22 08:56:29 |DEBUG  | PHP    Reloading PHP versions
[Web Server ] Nov 22 08:56:30 |DEBUG  | PHP    Using PHP version 7.4.9 (from default version in $PATH)
[Web Server ] Nov 22 08:56:30 |INFO   | PHP    listening path="C:\\wamp64\\bin\\php\\php7.4.9\\php-cgi.exe" php="7.4.9" port=57697
```

Dans un navigateur, on lance l'application en saisissant dans la barre d'adresse : localhost:8000 On obtient la page ci-dessous

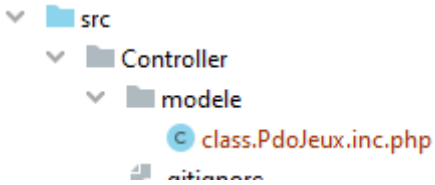


Le modèle

Le framework Symfony est un ensemble de composants. Dans cette version nous utiliserons le modèle de l'application existante, à savoir la classe d'accès aux données PdoJeux.

Dans le dossier src/Controller, il faut créer un nouveau dossier **modele**.

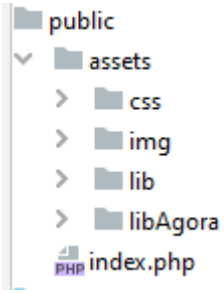
Copier le fichier existant class.PdoJeux.inc.php dans le dossier src/Controller
On obtient :



Dans le fichier .env à la racine du projet, ajouter les paramètres pour l'accès à la base de données. Dans l'ancienne application ces paramètres de configuration figuraient dans le fichier app/ _config.inc.php.

Les vues

On commence par copier le style et les scripts javascript. Création d'un dossier assets dans le dossier public et copie des dossiers se trouvant dans le dossier web de l'ancienne version d'AgoraBo.
On obtient :



Twig est le moteur de template préconisé par le framework Symfony. Comme il a été déjà utilisé dans l'application existante, les vues vont s'intégrer sans trop de modifications.

On supprime le fichier templates/base.html.twig . C'est le template de base (père) et on va le créer à partir de notre existant.

Copier les vues suivantes du projet existant dans le dossier templates de AgoraBoS :
accueil.html.twig
layout.html.twig
connexion.html.twig
lesGenres.html.twig
liste.html.twig
menu.html.twig

Renommer la vue layout.html.twig en base.html.twig. C'est le nom par défaut pour un template père dans Symfony 5.

Création des contrôleurs et des routes

Lorsqu'une requête HTTP arrive au serveur, Symfony essaie de trouver une *route* qui corresponde au *chemin de la requête*. Une *route* est le lien entre le chemin de la requête et une fonction devant créer la *réponse* HTTP associée à cette requête.
Cette fonction est nommée "contrôleur". Dans Symfony, la plupart des contrôleurs sont implémentés sous la forme de classes PHP. On peut créer ces classes manuellement, mais Symfony permet de nous aider à aller plus vite.